Merging Partial Behavioral Models

Arvind Easwaran, CIS700-03

Preliminaries

- Behavioral Model
  - Model describing behavioral aspect of a software system
  - Examples
    - State-based : Labeled Transition System (LTS)
    - Scenario-based : Message Sequence Charts (MSC)
- Complete Behavioral Model
  - Describes all possible behaviors of a software system

Motivation

- Construction of complete models is a complex task
- Partial behavioral models
  - Specified by different users with different viewpoints
  - Covering different components of a system
  - Multiple descriptions of the same component
  - Scenario based partial descriptions (MSCs)

State-based specifications

- State-based models are more amenable to verification
- Synthesis of state-based model from partial scenario specifications (LTS from MSC)
- LTS models are inherently absolute (disallow all transitions not explicitly shown in the model)
- But model absoluteness is limiting (partial scenarios)
- Requires state-based models which can explicitly model unknown(partial) behaviors
  - Modal Transition Systems (MTS), Partial LTS
  - MTS from MSC

Model Merging

- Analysis effective in state-based models describing complete behavior of system
- Justifies merging of partial models (merging MTSs for different scenarios)
- Problem
  - How to merge MTSs describing different, yet overlapping aspects of a system
  - How to combine MTSs of the same aspect specified by different users with different viewpoints

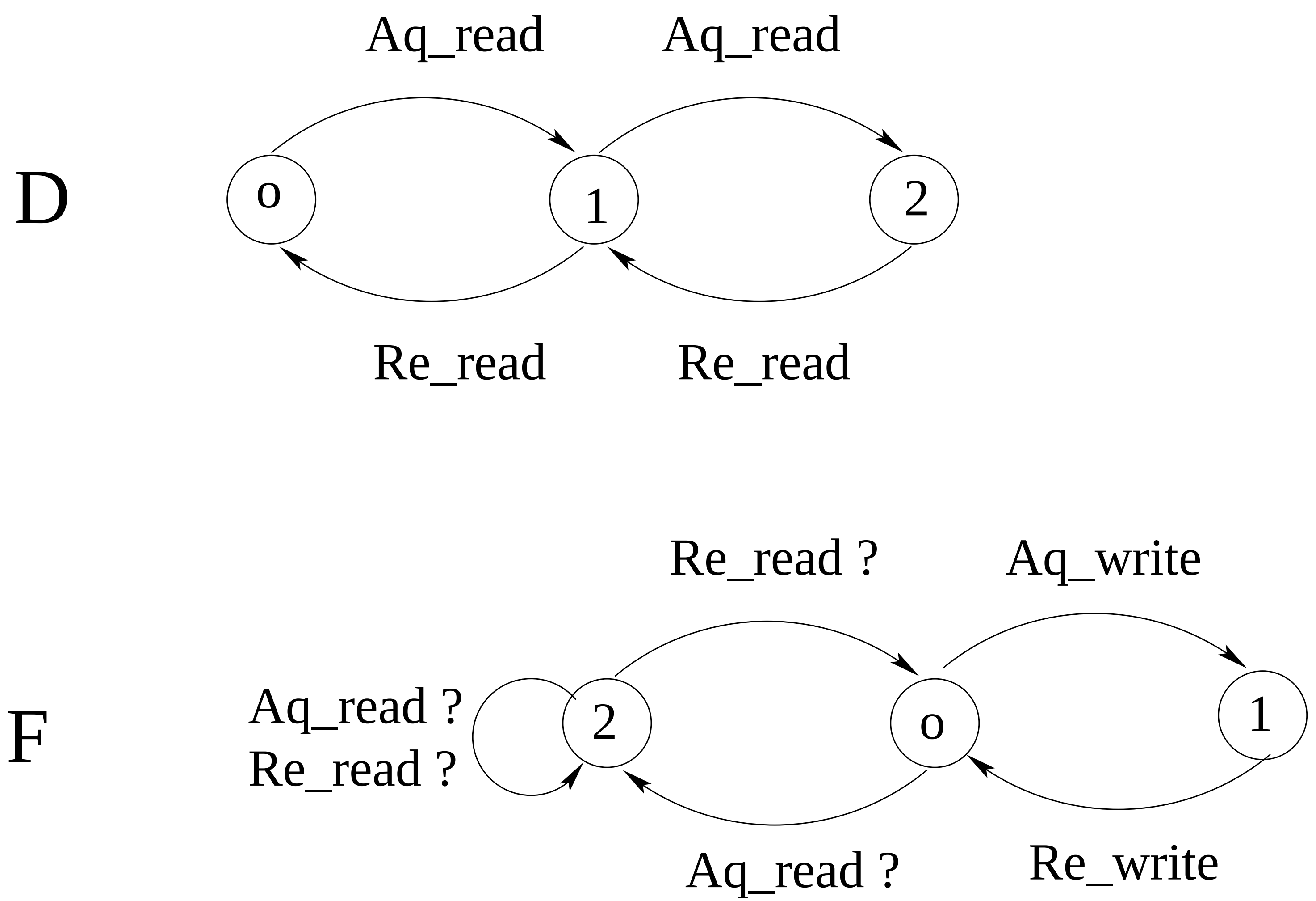
Labeled Transition System

- Act be a universal set of observable actions
- $Act_\tau = Act \cup \{\tau\}$ where τ is internal action
- Labeled Transition System(LTS)
  $P = \langle S, L, \Delta, s_0 \rangle$ where
  - S is a set of states
  - $L \subseteq Act_\tau$ is a set of action labels
  - $\Delta \subseteq (S \times L \times S)$ is a transition relation
  - $s_0 \in S$ is the initial state
- $\alpha P = L \setminus \{\tau\}$ denotes observable action set

Modal Transition System

- Modal Transition System(MTS) extends LTS with an additional set of uncertain transitions
- MTS $M = \langle S, L, \Delta^r, \Delta^p, s_0 \rangle$, $\Delta^r \subseteq \Delta^p$
- Δ^r represents required transitions and $\Delta^p \setminus \Delta^r$ represents maybe transitions
- LTS is a special type of MTS

Example



Aq_read
Aq_read
D
o
1
2
Re_read
Re_read
Re_read ?
Aq_write
Aq_read ?
Re_read ?
F
2
o
1
Aq_read ?
Re_write

MTS Semantics

- MTS $M = \langle S, L, \Delta^r, \Delta^p, s_0 \rangle$
- $M \longrightarrow_r^l M'$ if $M' = \langle S, L, \Delta^r, \Delta^p, s_0' \rangle$ and $(s_0, l, s_0') \in \Delta^r$
- $M \longrightarrow_m^l M'$ if $M' = \langle S, L, \Delta^r, \Delta^p, s_0' \rangle$ and $(s_0, l, s_0') \in \Delta^p \setminus \Delta^r$
- M proscribes l $(M \not\longrightarrow^l)$ if M cannot transit on l

Semantics Contd.

- $\omega = \omega_1 \cdots \omega_k \in Act_\tau$
- $(M \xrightarrow{\omega}_r N) \Rightarrow$
  - $\exists M_0, \cdots, M_k; M_0 = M, M_k = N$
  - $\forall i, (M_i \xrightarrow{\omega_{i+1}}_r M_{i+1}), 0 \leq i < k$
- $M \overset{l}{\Longrightarrow}_r M'$ denotes $M \xrightarrow{\tau^* l \tau^*}_r M'$

Semantics Contd.

- $(M \longrightarrow^{\omega}_{m} N) \Rightarrow$
  - $\exists M_0, \cdots, M_k;\ M_0 = M, M_k = N$
  - $\forall i, (M_i \longrightarrow^{\omega_{i+1}}_{p} M_{i+1}), 0 \leq i < k$
  - $\exists M_j, (M_j \longrightarrow^{\omega_{j+1}}_{m} M_{j+1}), 0 \leq j < k$
- $M \Longrightarrow^{l}_{m} M'$ denotes
  - $\exists M'', M \longrightarrow^{\tau^* l}_{m} M''$
  - $M'' \longrightarrow^{\tau *}_{r} M'$

MTS Refinement

- Refinement of a MTS results in a more concrete model than the original
- Some knowledge over maybe behavior is gained
- “More defined than” relation
- Intuitively, refinement converts some maybe transitions to required ones and some other maybe transitions are removed completely

Refinement Definition

- ρ be universe of MTSs
- $M \preceq N$ when $\alpha M = \alpha N$ and
  - (M, N) contained in some refinement relation $R \subseteq \rho \times \rho$
  - $\forall l \in Act_\tau$,
    1. $(M \longrightarrow^{l}_{r} M') \Rightarrow$
       $((\exists N', N \longrightarrow^{l}_{r} N') \wedge (M', N') \in R)$
    2. $(N \longrightarrow^{l}_{p} N') \Rightarrow$
       $((\exists M', M \longrightarrow^{l}_{p} M') \wedge (M', N') \in R)$

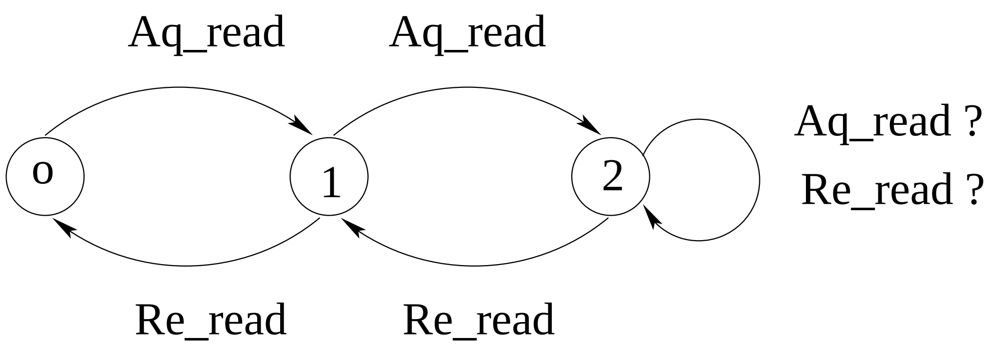
Label Hiding

- Refinement requires alphabets of models to be same
- Hiding makes set of actions of a model unobservable to environment
- All transitions labeled with the hidden action are replaced with τ
- $M@\alpha X$ denotes MTS with label set X
    - All labels not in X are replaced with τ

Observational Refinement (OR)

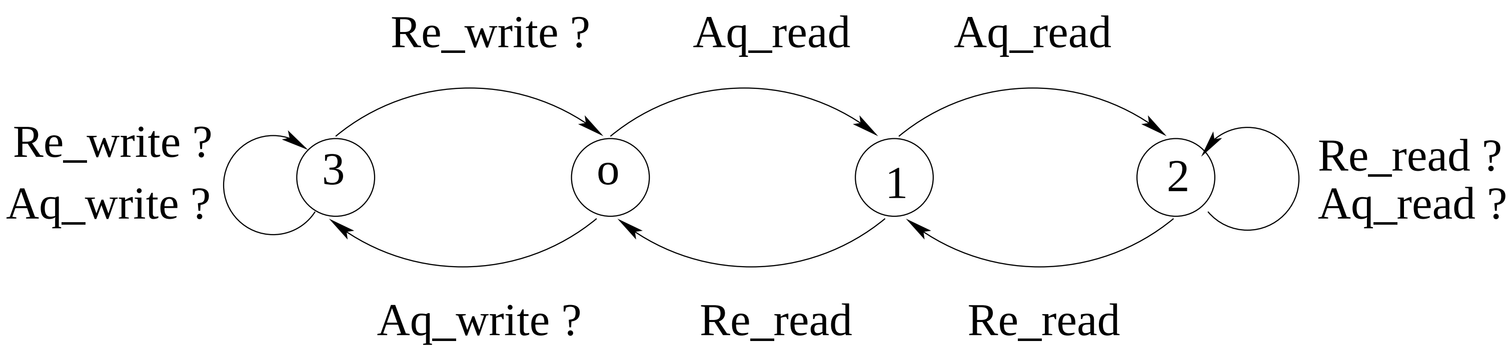
- $M \preceq_o N$ when $\alpha M = \alpha N$ and
  - (M, N) is contained in some refinement relation $R \subseteq \rho \times \rho$
  - $\forall l \in Act$,
    - $(M \Longrightarrow^l_r M') \Rightarrow$ $((\exists N', N \Longrightarrow^l_r N') \wedge (M', N') \in R)$
    - $(N \Longrightarrow^l_p N') \Rightarrow$ $((\exists M', M \Longrightarrow^l_p M') \wedge (M', N') \in R)$

Example



Aq_read
Aq_read
Aq_read ?
Re_read ?
0
1
2
Re_read
Re_read

D



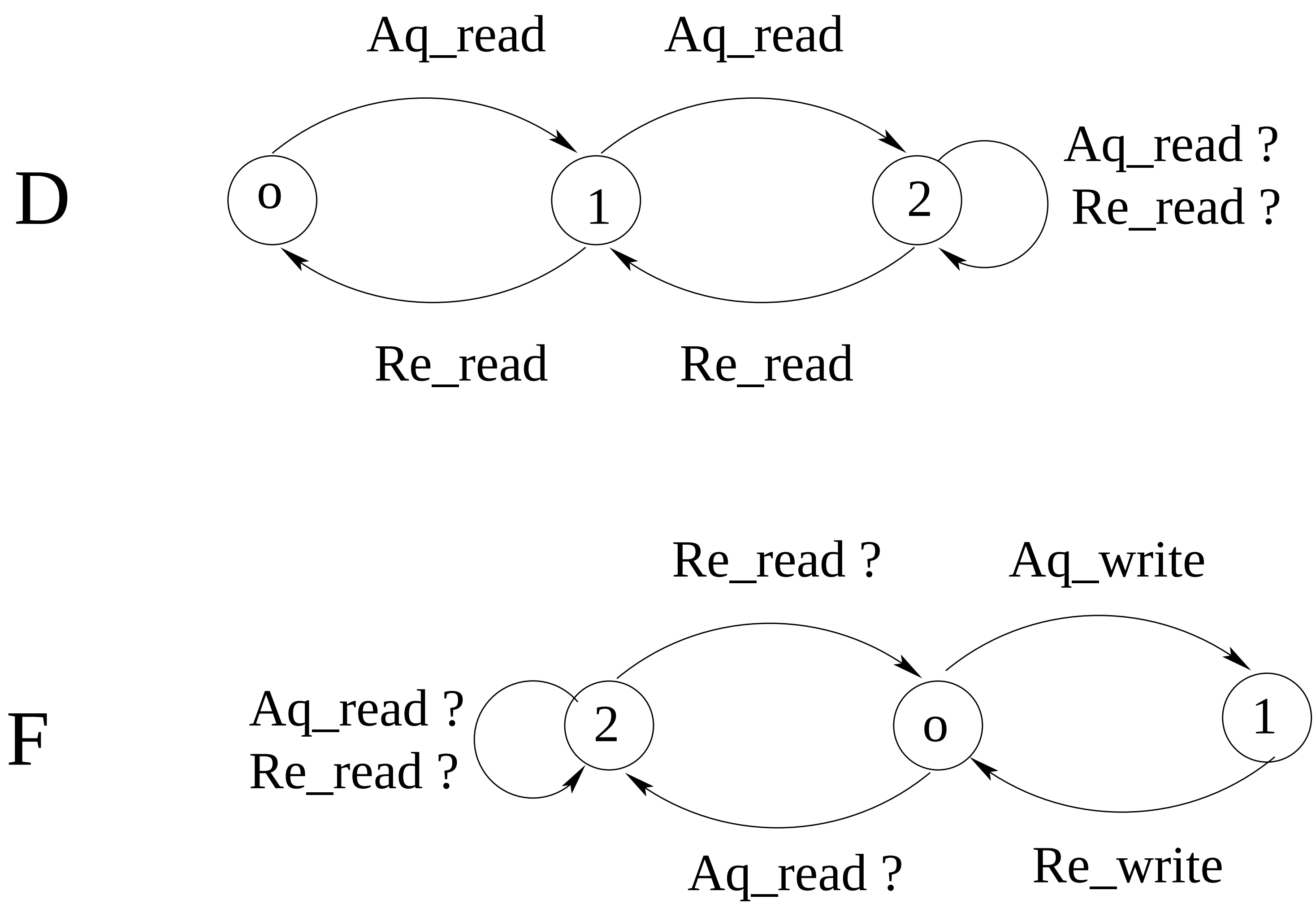
Re_write ?
Aq_read
Aq_read
Re_write ?
Aq_write ?
3
0
1
2
Re_read ?
Aq_read ?
Aq_write ?
Re_read
Re_read

E

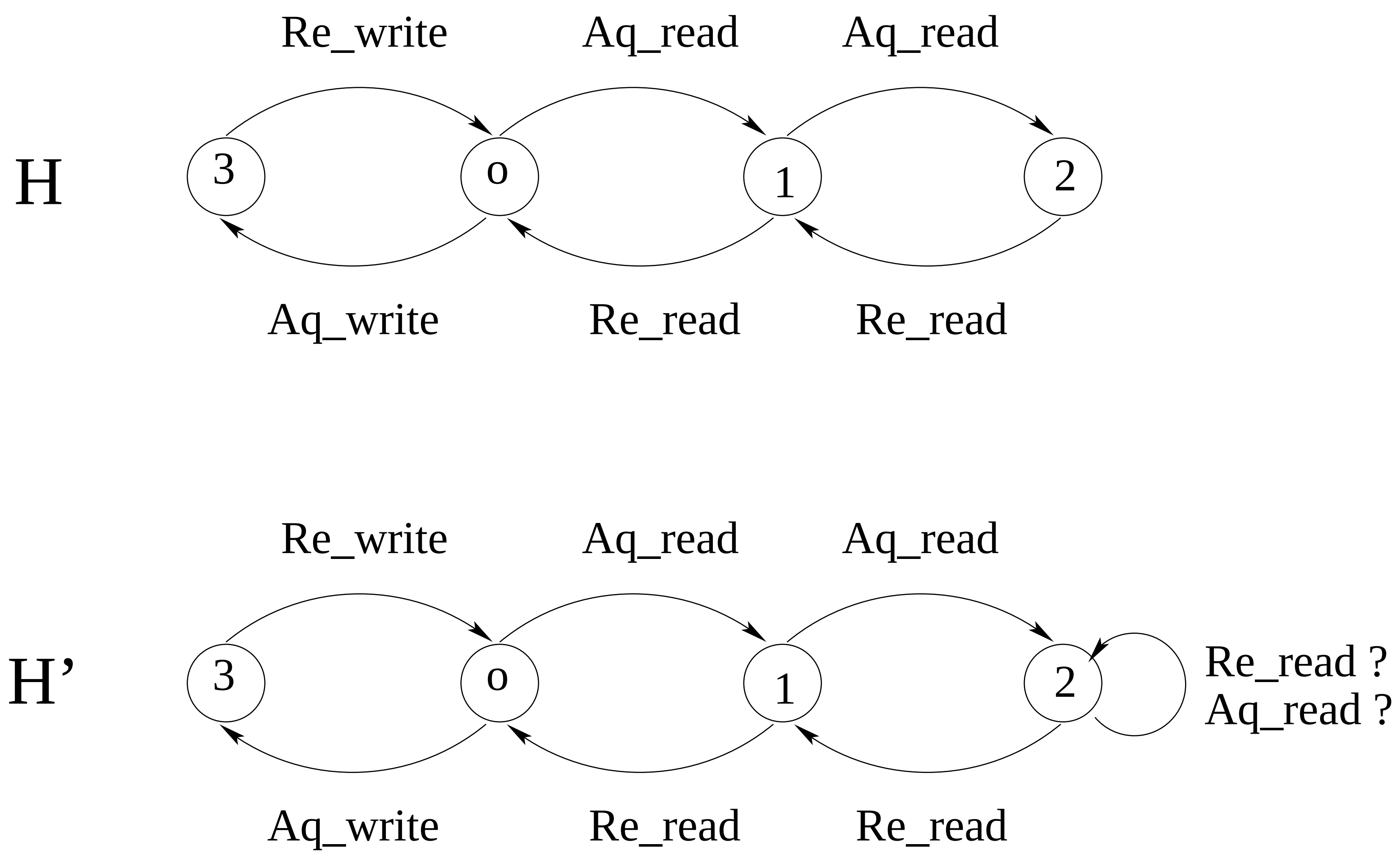
MTS Merging

- Knowledge from two partial models(MTS) used to generate a unified MTS
- Merging is about finding a common refinement of the two models
- Models being merged can have different action labels
- P is a common observational refinement of M and N if $(\alpha P \supseteq (\alpha M \cup \alpha N))$, $(M \preceq_o P@\alpha M)$ and $(N \preceq_o P@\alpha N)$

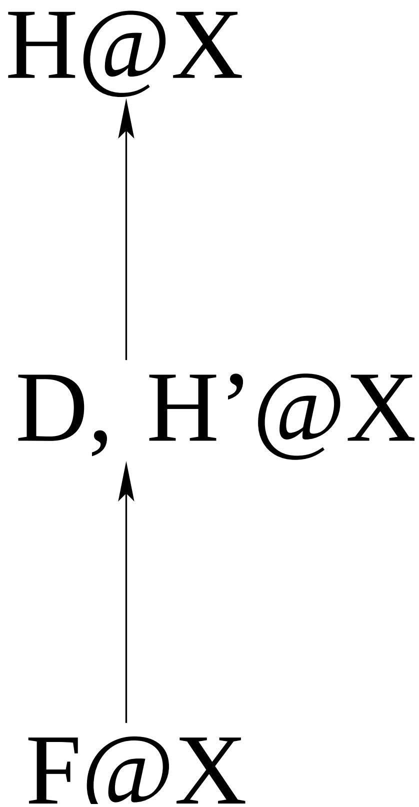
Example


D
Aq_read
Aq_read
o
1
2
Aq_read ?
Re_read ?
Re_read
Re_read
F
Re_read ?
Aq_write
Aq_read ?
Re_read ?
2
o
1
Aq_read ?
Re_write

Example Contd.



Example Contd.



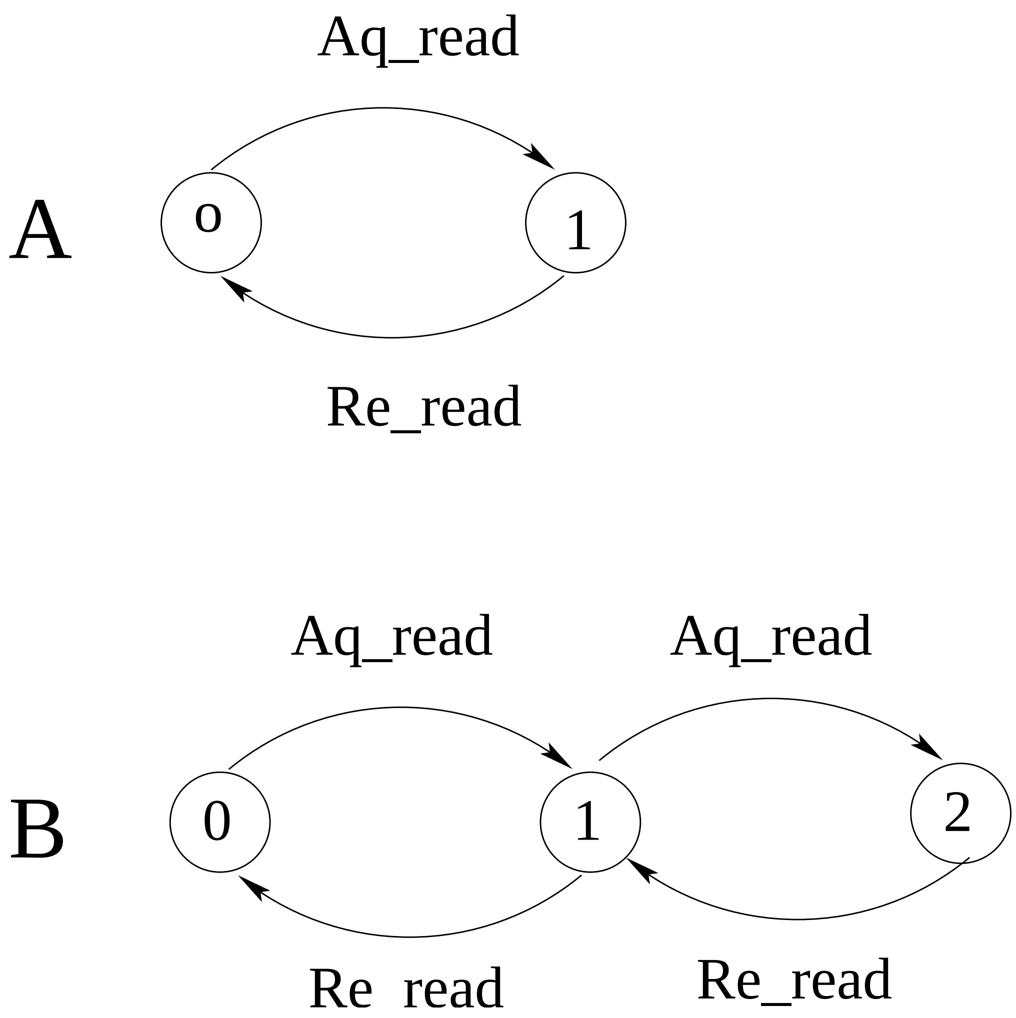
Least Common Refinement

- H and H' are both common refinements of D and F
- H' is the preferred common refinement; H proscribes three or more readers which is not required
- P is the least common refinement(LCR) of M and N if P is a common refinement of M and N, $\alpha P = (\alpha M \cup \alpha N)$, and for any common refinement Q of M and N, $P \preceq_o Q@\alpha P$
- But common refinement or LCR need not exist for two MTSs

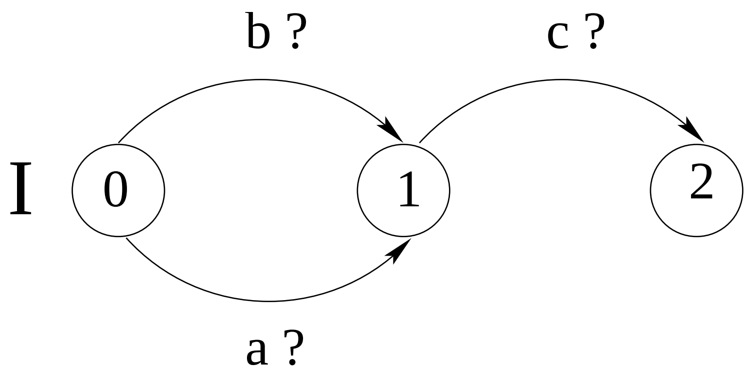
Model Consistency

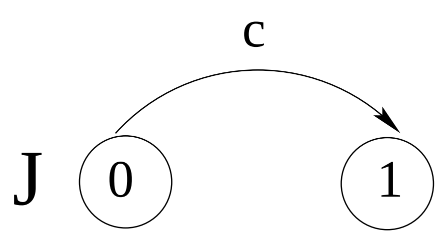
- Two MTSs M and N are consistent if and only if there exists an MTS P such that P is a common refinement of M and N
- Consistency does not guarantee the existence of LCR
- An MTS P is minimal common refinement (MCR) of M and N if P is a common refinement of M and N, $\alpha P = (\alpha M \cup \alpha N)$, and there is no MTS $Q \not\equiv P$ such that Q is a common refinement of M and N and $Q@\alpha P \preceq_o P$

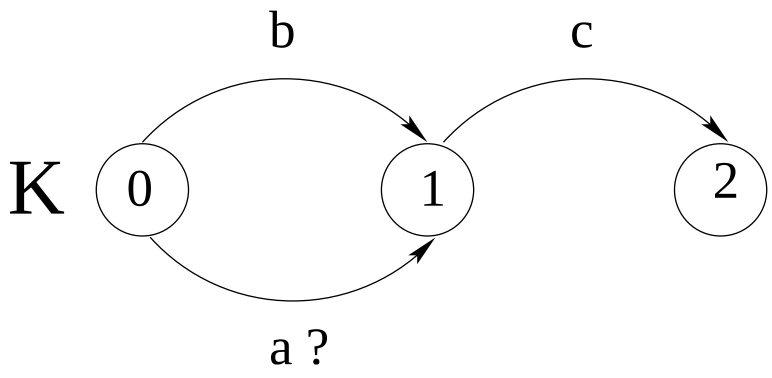
Example

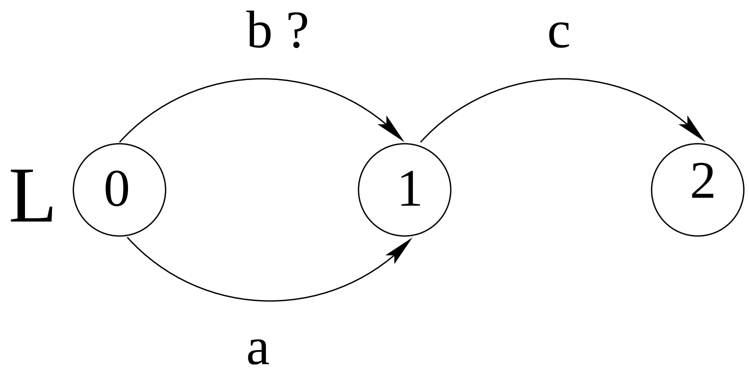


Example



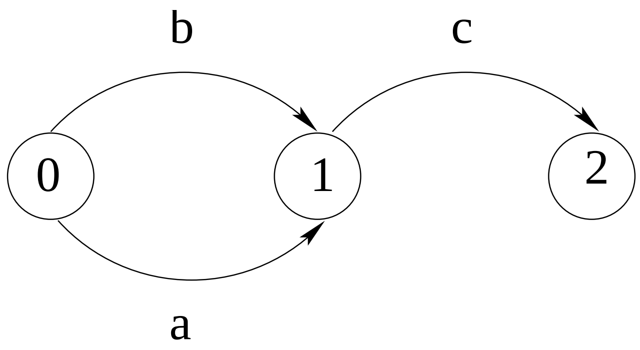


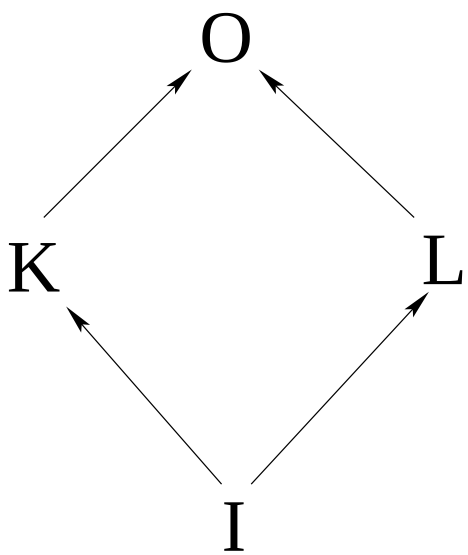


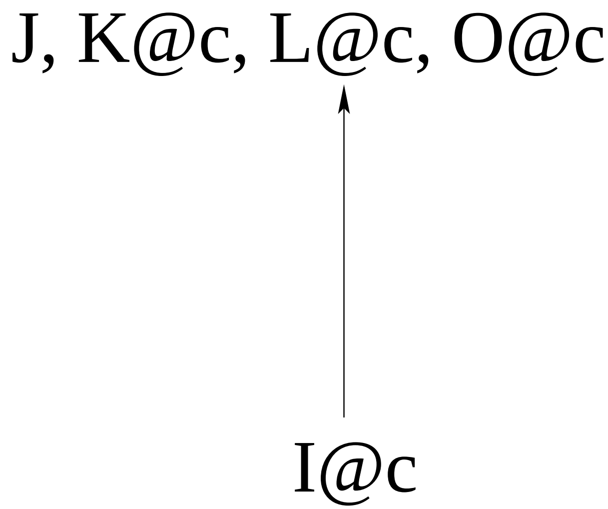


Example Contd.

O





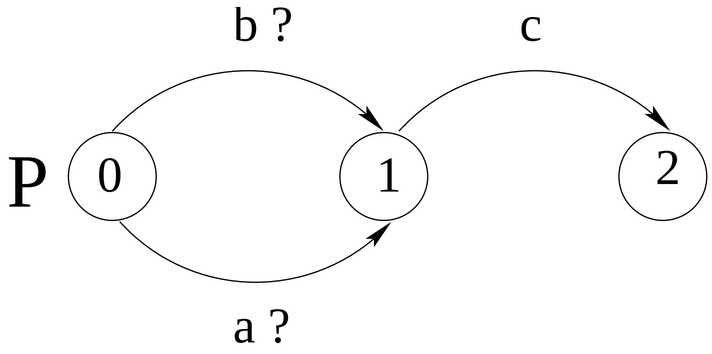


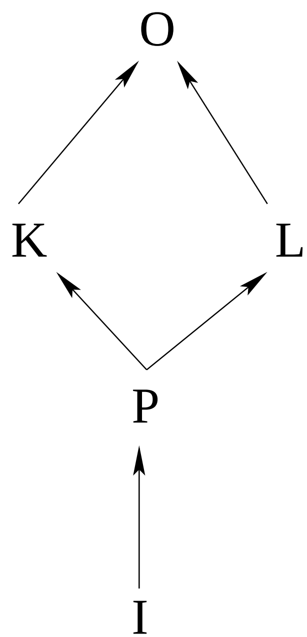
Greatest Lower Bound

- Merging consistent models with no LCR will result in any one of the MCRs
- A better approach would be to find the greatest lower bound(glb) of all MCRs
- The user can then build one of the MCRs using this glb model
- glb is unique with respect to observational equivalence

- glb always exists
- glb itself might not be a common refinement of the models being merged
- Let M and N be consistent. Q is a lower bound of all MCRs if $\alpha Q = (\alpha M \cup \alpha N)$ and for any MCR P, it holds that $Q \preceq_o P$. Q is a glb if for any other lower bound Q', it holds that $Q' \preceq_o Q$
- If P is a LCR, then P is also the glb of all MCRs of M and N

Example





Algorithms

- Consistency checking between two partial models
- Constructing LCR if it exists
- Supporting construction of MCRs using glb
- $+_u$ Operator
  - Used for consistency checking
  - Gives a upper bound for all MCRs
- $+_l$ Operator
  - Gives a lower bound (approximate glb)
  - Used to construct the LCR or one of the MCRs

$+_u$ Operator

- **TD** $\forall l \notin \alpha N \; (M \stackrel{l}{\longrightarrow}_r M') \Rightarrow (M +_u N \stackrel{l}{\longrightarrow}_r M' +_u N)$
- **TM** $\forall l \notin \tau \; (M \stackrel{l}{\longrightarrow}_r M') \wedge (N \stackrel{l}{\longrightarrow}_m N') \Rightarrow (M +_u N \stackrel{l}{\longrightarrow}_r M' +_u N')$
- **MD** $\forall l \notin \alpha N \; (M \stackrel{l}{\longrightarrow}_m M') \Rightarrow (M +_u N \stackrel{l}{\longrightarrow}_r M' +_u N)$
- **TT** $\forall l \notin \tau \; (M \stackrel{l}{\longrightarrow}_r M') \wedge (N \stackrel{l}{\longrightarrow}_r N') \Rightarrow (M +_u N \stackrel{l}{\longrightarrow}_r M' +_u N')$
- **MM** $\forall l \notin \tau \; (M \stackrel{l}{\longrightarrow}_m M') \wedge (N \stackrel{l}{\longrightarrow}_m N') \Rightarrow (M +_u N \stackrel{l}{\longrightarrow}_m M' +_u N')$

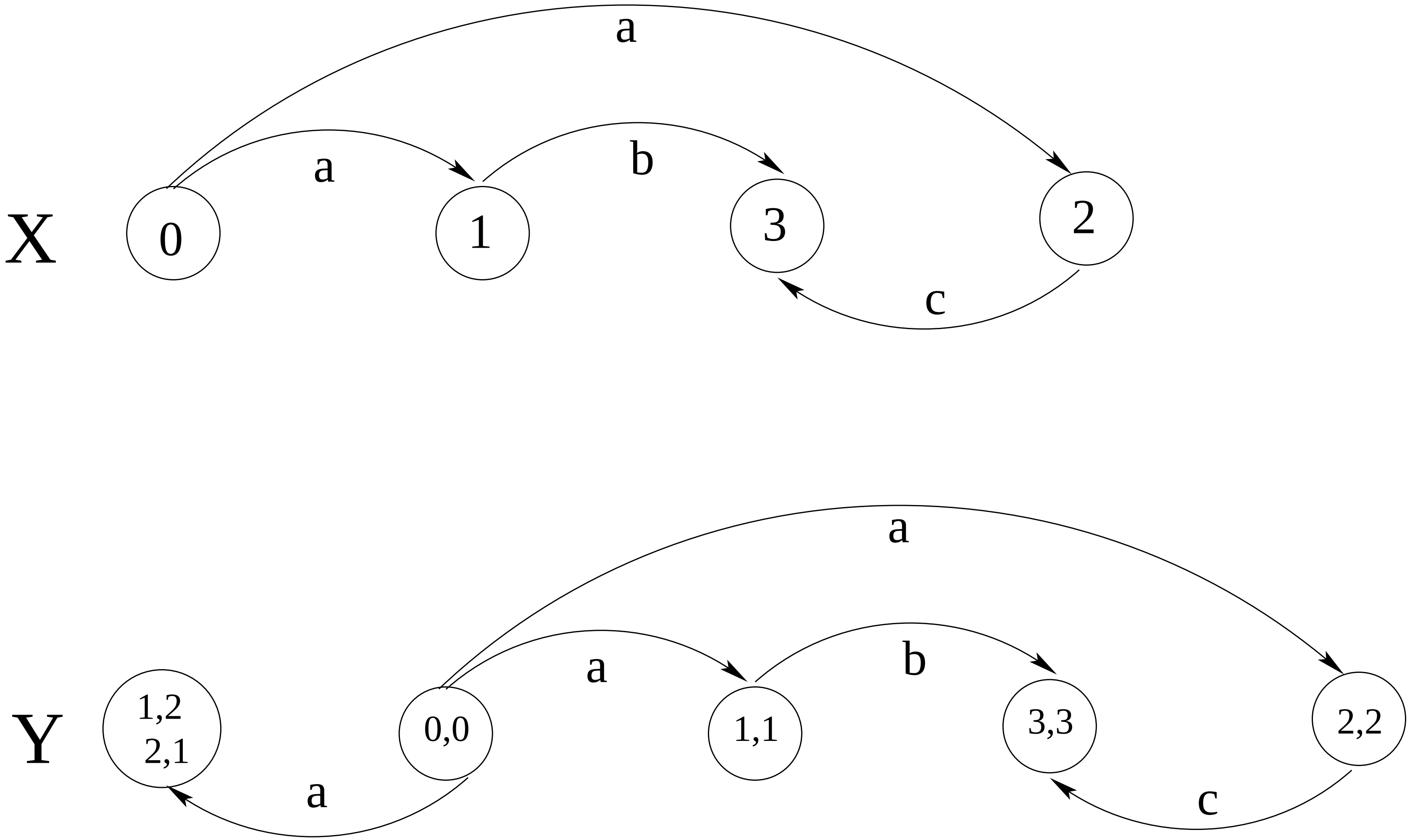
Disagreement states

- $M = \langle S_M, L_M, \Delta^r_M, \Delta^p_M, s_{0_M} \rangle$
- $N = \langle S_N, L_N, \Delta^r_N, \Delta^p_N, s_{0_N} \rangle$
- $(m, n) \in (S_M \times S_N)$ is a disagreement state if $\exists l \in (\alpha M \cap \alpha N)$ such that
- $M_m \overset{l}{\longrightarrow}_r$ and $N_n \overset{l}{\not\longrightarrow}$ or
- $M_m \overset{l}{\not\longrightarrow}$ and $N_n \overset{l}{\longrightarrow}_r$
- Consistent models ensure disagreement states can progress using unobservable actions

Determinacy Condition

- $C = \langle S_M \times S_N, L_C, \Delta^r_C, \Delta^p_C, (s_{0_m}, s_{0_n}) \rangle$
- Determinacy condition holds if $\forall (m, n) \in C$ and all $l \in L_M \cap L_N$ it is not the case that M_m and N_n are non-deterministic on l
- Consistent M and N, $(M +_u N)$ satisfying determinacy $\Rightarrow$
  - $M +_u N$ is a common observational refinement
  - For every Q that is a MCR, $Q@\alpha(M +_u N) \preceq_o (M +_u N)$

Example



Consistency Checker

Algorithm : Consistency check

- INPUT: MTSs M and N
- OUTPUT: If M and N not consistent, return one of the disagreement states else return null

- Build $M +_u N$ marking disagreement states
- For each marked state (m, n)
- If $N_n \not\longrightarrow^{l}$
- If $\forall \omega \in (Act_\tau \setminus \alpha M)^*, N_n \not\longrightarrow^{\omega l}$
- Return (m, n)
- Else if $M_m \not\longrightarrow^{l}$ **Similar as above**
- Else return null

$+_l$ Operator

- DM $\forall l \notin \alpha M \; (N \xrightarrow{l}_m N') \Rightarrow$
  $((M +_l N) \xrightarrow{l}_m M +_l N')$
- MD $\forall l \notin \alpha N \; (M \xrightarrow{l}_m M') \Rightarrow$
  $((M +_l N) \xrightarrow{l}_m M' +_l N)$
- If M and N are consistent and $(M +_l N)$ satisfies the determinacy condition, then for any MCR Q of M and N, $(M +_l N) \preceq_o Q@\alpha(M +_l N)$
- $(M +_l N)$ approximates the glb of M and N

DM and MD rules

- To obtain exact glb, DM and MD rules should convert some maybe transitions into required transitions
- If all are converted we get $M +_u N$
- If none are converted we get $M +_l N$
- If DM and MD rules are never applied then $+_u \equiv +_l$ and they produce LCR

Elaboration

- Refinement of lower bound obtained using $+_l$ into a MCR
- **Algorithm : Elaboration**
- INPUT: MTSs M and N; consistent and satisfy determinacy
- OUTPUT: MTS P which is the required MCR/LCR

- Build $P = M +_l N$ marking disagreement states
- For each marked state (m,n) if $N \not\longrightarrow^l$
- Build $T = \{\omega \in (\alpha N \setminus \alpha M)^* : \exists N', (N_n \Longrightarrow^{\omega}_{m} N'_n) \wedge (N'_n \Longrightarrow^{l}_{m} N''_n)\}$
- User chooses $\omega' \in T$ (If $|T| = 1$ we get LCR)
- Replace maybe transitions with required ones; $(M_m +_l N_n) \Longrightarrow^{\omega}_{r} (M_m +_l N'_n)$
- Else if $M_m \not\longrightarrow^l$ **similar as above**

Complexity Analysis

- S_M and S_N are states of M and N
- T_M and T_N are transitions; T_i is $O(S_i \times L_i)$
- Potential size of state space of common refinement is $S = O(|S_M| \times |S_N|)$
- Consistency check is similar to weak bisimulation $O(L \times S \times T)$
- Computing $+_u$ and $+_l$ does not increase this complexity
- Use BFS for computing T in the elaboration algorithm

References

- “Merging Partial Behavioral Models”, Sebastian Uchitel and Marsha Chechik, ACM International Symposium on Foundations of Software Engineering, 2004
- “Synthesis of Behavioral Models from Scenarios”, Sebastian Uchitel, Jeff Kramer and Jeff Magee, IEEE Transactions on Software Engineering, Feb 2003
- “From Scenarios to Timed Automata: Building Specifications from User Requirements”, Stephane Some, Rachida Dssouli and Jean Vaucher, Asia Pacific Software Engineering Conference, 1995